



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/635,956	08/10/2000	Timothy C. Loose	47079-00058	6262

30223 7590 09/23/2003  
JENKENS & GILCHRIST, P.C.  
225 WEST WASHINGTON  
SUITE 2600  
CHICAGO, IL 60606

EXAMINER

COBURN, CORBETT B

ART UNIT	PAPER NUMBER
----------	--------------

3714

14

DATE MAILED: 09/23/2003

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

**MAILED**  
**SEP 22 2003**  
**GROUP 3700**

Commissioner for Patents  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

**BEFORE THE BOARD OF PATENT APPEALS  
AND INTERFERENCES**

Paper No. 14

Application Number: 09/635,956  
Filing Date: August 10, 2000  
Appellant(s): LOOSE, TIMOTHY C.

---

Timothy Kowalski  
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 23 June 2003.

**(1) *Real Party in Interest***

A statement identifying the real party in interest is contained in the brief.

Art Unit: 3714

**(2) *Related Appeals and Interferences***

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

**(3) *Status of Claims***

The statement of the status of the claims contained in the brief is correct.

**(4) *Status of Amendments After Final***

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) *Summary of Invention***

The summary of invention contained in the brief is correct.

**(6) *Issues***

The appellant's statement of the issues in the brief is correct.

**(7) *Grouping of Claims***

While the Grouping of the Claims section fails to state that the claims do not stand or fall together, it appears from the Appellant's Brief that this is what is intended. Appellant has divided the claims into subgroups for discussion, with these subgroups standing and falling together.

**(8) *Claims Appealed***

The copy of the appealed claims contained in the Appendix to the brief is correct.

Art Unit: 3714

**(9) Prior Art of Record**

The following is a listing of the prior art of record relied upon in the rejection of claims under appeal.

6,394,900	McGlone et al.	05-2002
6,315,663	Sakamoto	11-2001

**(10) Grounds of Rejection**

The ground(s) of rejection contained in paper 7 and which are maintained and repeated herein, are applicable to the appealed claims:

1. Claims 1-5 & 7-28 rejected under 35 U.S.C. 103(a) as obvious over McGlone et al. (US Patent Number 6,394,900)

**Claims 1, 9, 20:** McGlone teaches a slot machine with a central processing unit (422) for operating the slot machine in response to a wager. There is a reel mechanism including a motor (38) a symbol bearing reel (32) and a reel driver (402). The motor has a rotatable shaft upon which the reel is mounted. (Fig 1c) The reel driver includes a local microcontroller (612) distinct from and serially coupled to the CPU. (Fig 6) The reel driver is coupled to the motor to cause the motor to rotate the reel. (Abstract) The reel driver performs low-level reel driver operations independent from the CPU. (Fig 8) The CPU issues high-level commands to the reel driver related to the rotation of the reel. (Abstract) McGlone also teaches that the master gaming controller includes a memory storing software for device drivers for at least some of the slot reel peripherals.

Art Unit: 3714

(Col 3, 35-41) These device drivers are configuration data and are sent to the local microprocessor for configuring it to a reel spinning game conducted with a slot machine.

Furthermore, McGlone teaches that, "The peripheral controller may have a non-volatile memory arranged to store configuration parameters specific to the slot reel peripheral and state history information of the slot reel peripheral. In one embodiment, the non-volatile memory might be used to store the configuration parameters needed to drive the slot reel using the drive mechanism including a moment of inertia of the slot reel, the size of the slot reel and one or more acceleration parameters." (Col 3, 17-24) The suggestion that the peripheral (i.e., local) controller may have non-volatile memory for storing configuration data provides a strong suggestion that the opposite may also be true. If the peripheral controller does not have non-volatile memory for storing configuration data, that data must be loaded to the peripheral controller from the central procession unit. Doing this would eliminate unnecessary duplication of parts because the data could be stored in one set of non-volatile memory instead of on non-volatile memory associated with each peripheral controller. This would reduce the cost of the gaming machine. It would have been obvious to one of ordinary skill in the art at the time of the invention to have the central processing unit send configuration data to the local microcontroller for configuring the microcontroller to a reel spinning game conducted with a slot machine in order to eliminated unnecessary duplication of parts, thus reducing costs.

**Claim 2:** Slot machines inherently stop the symbols in visual association with one or more paylines.

**Claim 3:** The low-level reel driver operations include monitoring the reel and at least partially controlling its position. (Abstract)

**Claims 4, 11:** The local microcontroller monitors the reel by sampling its state multiple times per second in real time and responds with control commands for controlling the position of the reel. (Col 8, 45-57)

**Claim 5:** The local microcontroller is serially connected to the CPU via a Universal Serial Bus. (Col 2, 44-48)

**Claims 7, 10:** The CPU issues high-level commands to the local microcontroller. The high-level commands include a start reel command for starting the reel and a stop reel command for stopping the reel. (Col 7, 25-33, Col 8, 28-32)

**Claim 8, 13:** The reel includes an encoder (barcode) for indicating the position of the reel. The reel driver includes a barcode reader (408) that is coupled to the microcontroller so that the microcontroller may monitor the position of the reel. (Col 8, 45-57)

**Claim 12:** Claim 12 is merely a restatement of claims 1, 4, and 7, which see.

**Claims 14, 16, 18, 21:** McGlone teaches that the configuration data includes “parameters needed to drive the slot reel using the drive mechanism, including the moment of inertia of the slot reel, the size of the slot reel and one or more acceleration parameters.” (Col 3, 21-24)

**Claims 15, 17, 19, 22, 26, 27:** As noted above, McGlone suggests transmission of configuration data by the CPU to the local microcontroller and processing of that data by the local microprocessor. McGlone also teaches communication between the local

Art Unit: 3714

microcontroller and the CPU. (Col 3, 32-33) While McGlone fails to specifically teach communication of the status of configuration from the local processor to the CPU, it would be obvious to do so. If the local controller were misconfigured, errors could occur in the game. These errors could result in players winning when they should not win and losing when they should not lose. Neither of these conditions is acceptable to a casino. For example, if the configuration data were for the wrong type of position decoder, it would be impossible to determine which symbols appear along a payline. It would have been obvious to one of ordinary skill in the art at the time of the invention to have communicated of the status of configuration from the local processor to the CPU and to have compared the configuration data with the determined type of position encoder in order to prevent the game machine from operating in a misconfigured condition that would lead to errors in the game.

**Claim 23:** McGlone teaches providing a physical symbol-bearing reel (Fig 1b) including an encoder (410) for indicating the position of the reel. (Col 1, 43-44) McGlone teaches providing a reel controller for performing low-level operations related to the movement of the reel. (Fig 8) There is also a CPU for issuing high-level commands to the reel controller related to the movement of the reels. (Abstract) McGlone also teaches that the microcontroller that operates the reels must have the parameters and operation features of the position sensor in order to operate the reels. (Col 1, 42-48, Col 2, 66 – Col 3, 4) The peripheral controller receives device drivers for operating some of the peripherals from the CPU. (Col 3, 34-41) This would include the position encoder since the position encoder is necessary for the operation the slot machine. In order to determine which of

Art Unit: 3714

the various device drivers to download, it would be necessary for the reel controller to determine the type of encoder present and to report it to the CPU. It would have been obvious to one of ordinary skill in the art at the time of the invention to have the CPU send a command to the reel controller to determine the type of position encoder present in order to download the correct device drivers to the reel controller.

**Claim 24:** McGlone teaches encoders with one or more flags (i.e., tabs). (Col 8, 52-54)

Thus the type of the encoder is based on the number of flags on the encoder.

**Claim 25:** As pointed out in connection with claim 23 above, in order to download the correct device driver, it is necessary to determine the type of encoder being used.

**Claim 28:** As pointed out in connection with claim 24 above, there are different types of position encoders. The number of flags or tabs on the encoder determines the type of the encoder. As pointed out in connection with claim 23, the machine must determine the type of encoder present in order to correctly configure the device by downloading the correct device drivers. The easiest way to determine the type of encoder present would be to cause the motor to spin and count the number of flags on the encoder. Thus it would have been obvious to one of ordinary skill in the art at the time of the invention to have caused the motor to spin the reel and detect the physical characteristics (i.e., the number of flags) of the encoder in order to determine which type of encoder was present, thus enabling the loading of the correct device driver.

2. Claim 29 is rejected under 35 U.S.C. 103(a) as being unpatentable over McGlone in view of Sakamoto (US Patent Number 6,315,663).



**Claim 29:** McGlone teaches providing a physical symbol-bearing reel (Fig 1b) including an encoder (410) for indicating the position of the reel. (Col 1, 43-44) McGlone teaches providing a reel controller for performing low-level operations related to the movement of the reel. (Fig 8) There is also a CPU for issuing high-level commands to the reel controller related to the movement of the reels. (Abstract) McGlone teaches that the reel controller has “configuration parameters needed to drive the slot machine using the drive mechanism including a moment of inertia, the size of the slot reel and one or more acceleration parameters.” (Col 3, 21-24) But McGlone does not teach an acceleration or deceleration profile for accelerating or decelerating the reel. Sakamoto teaches an acceleration or deceleration profile for accelerating or decelerating the reel. (Col 12, 40-61) Having the reels accelerate and decelerate at varying speeds add visual interest to the slot machine game. It would have been obvious to one of ordinary skill in the art at the time of the invention to have the CPU send high-level commands concerning acceleration or deceleration profile for accelerating or decelerating the reel to the reel controller in order to add visual interest to the slot machine game.

**(11) Response to Argument**

Note: Examiner has answered the Appellant's arguments in the order presented and has maintained Appellant's paragraph numbering for ease of reference.

- I. Appellant has set forth a brief description of the Law of Obviousness. Since these are not arguments, Examiner will not address this section further.
- II. Appellant states that claims 1-5, 7-23 and 27 are patentable over McGlone. Appellant appears to have misconstrued both McGlone and the rejection. The rejection states that the

Art Unit: 3714

limitations, including the limitation concerning downloading of configuration data from the CPU to the reel controller, are met by McGlone. McGlone clearly discloses a preferred embodiment that downloads device drivers from the CPU to the reel controller. However, since the McGlone disclosure does not specifically state that the device drivers contain configuration data, and being wary of the use of inherency in formulating rejections, Examiner chose to err on the side of caution and reject the claim under 35 USC §103(a) by stating that device drivers obviously contain configuration data. To support this conclusion, Examiner pointed out in the specification where McGlone discloses that, in an alternate embodiment, configuration data may be stored on non-volatile memory on the reel controller. Configuration data is absolutely essential to the functioning of McGlone's reel controller.

A. Appellant argues that McGlone does not disclose, teach, or suggest a CPU that sends configuration data to a local microcontroller for configuring the local microcontroller.

Appellant also argues that McGlone teaches away from the CPU sending such data to the local microcontroller. As discussed above, Appellant appears to misconstrue both McGlone and the rejection.

1. McGlone clearly teaches that, in the preferred embodiment, the master gaming controller includes a memory storing software for device drivers (i.e., configuration data) for at least some of the slot reel peripherals. (Col 3, 35-41) McGlone also teaches, "The peripheral controller may have a non-volatile memory arranged to store configuration parameters specific to the slot reel peripheral and state history information of the slot reel peripheral. In one embodiment, the non-volatile memory might be used to store the configuration

Art Unit: 3714

parameters needed to drive the slot reel using the drive mechanism including a moment of inertia of the slot reel, the size of the slot reel and one or more acceleration parameters.” (Emphasis added.) (Col 3, 17-24) Thus, McGlone discloses two embodiments – one with non-volatile memory and a preferred embodiment without non-volatile memory. In the preferred embodiment, the configuration data is loaded to the reel controller by the master game controller or CPU.

There is, however, one minor problem – McGlone doesn’t use the words “configuration data” in connection with this embodiment. Instead, McGlone speaks in terms of “device drivers”. While one of ordinary skill in the art would know that device drivers must include “configuration data” (see paragraph IIB below), McGlone states in unambiguous terms that “configuration parameters” may be stored on the non-volatile memory of the second embodiment. Because such data is necessary for the functioning of McGlone’s device, it would have been obvious to one of ordinary skill in the art at the time of the invention to have downloaded the “configuration parameters” to the preferred embodiment.

2. Appellant argues that McGlone “teaches away” from the notion of loading the configuration data from the CPU. This is not the case. McGlone teaches two embodiments – one that loads configuration data from the CPU and one that does not. This cannot “teach away” from loading configuration data to the reel controller from the CPU.

3. Appellant discusses advantages to the Appellant's disclosed invention. This has no bearing on the issue of obviousness.
4. Appellant argues that the proposed modification of McGlone renders McGlone "unsatisfactory for its intended purpose". This is not the case. First, there is no actual modification to McGlone being suggested. McGlone discloses loading device drivers to the reel controller from the CPU as the preferred embodiment. The "suggested modification" is merely intended to clarify that McGlone recognizes that configuration data is needed to operate the reel controller and that, while McGlone speaks in terms of "device drivers" in reference to the preferred embodiment, these device drivers must obviously contain the configuration data discussed in connection with the second embodiment.

Furthermore, even if there were a modification, it would not render McGlone "unsatisfactory for its intended purpose." McGlone teaches the intended purpose of the invention in Col 2, lines 15-39:

"Disadvantages of the current slot machine architecture include the following. First, the number of types of motherboards needed to accommodate all of the potential combinations of gaming devices has become large. Second, the computational capabilities of the motherboard needed to drive all the devices has become large. Third, when devices are added to augment the features of the gaming machine or when devices are replaced for maintenance the steps necessary to rewire the device onto the motherboard and load the appropriate software onto the motherboard can be time consuming and require significant shutdown time for the gaming machine. Accordingly, it would be desirable to provide slot reels that are compatible with a standard communication protocol and/or connection system for installing or removing devices controlled by a master gaming controller. A slot reel gaming peripheral that is

compatible with a standard communication protocol and/or connection system may reduce the number of types of motherboards that are needed for the gaming machine and may reduce the amount of maintenance time when a slot reel is replaced. Further, it would be desirable to have the slot reel gaming peripheral control some of its own functions rather than having all the functions controlled by the master gaming controller. This feature might reduce the load on the computational resources of the master gaming controller. “ (Emphasis added)

The “suggested modification” to McGlone provides “slot reels that are compatible with a standard communication protocol and/or connection system for installing or removing devices controlled by a master gaming controller.” The advantages enumerated by McGlone (reduction of the number of types of motherboards, reduction of the amount of maintenance time when reels are replaced, and reduction of the computational load of the master game controller) are still achieved. Therefore, the proposed modification does not render McGlone “unsatisfactory for its intended purpose” or “change the principle of operation” of the reference.

B. Appellant argues that device drivers are not “configuration data”. Appellant is incorrect.

1. Appellant argues that device drivers are programs and not data. This shows a misconception of what data is. Data is information. A program is information. This has long been recognized by practitioners of the art. Device drivers are an excellent example of a program that bears information – in this case, configuration data. As noted in *Microcomputers* (Sandon, IBM Microelectronics Division, 1999, <http://www.mrw.interscience.wiley.com/eeee/63/1663/W.1663->

[4.html](#)), “The device driver is the piece of code that does have device specific information.” (See Appendix A)

A device driver provides information concerning the characteristics of a device and uses that information to control the device. It is important to note that the device driver could not possibly function without configuration data.

Configuration data must be part of the device driver.

A device driver for a slot machine reel controller, for instance, would have to contain information concerning the number of symbols of a reel, the number of steps per revolution of a stepper motor, number of pulses per step, etc. in order to function. The device driver uses this information to control the reels and could not control the reels without it.

2. Appellant’s argument that Examiner changes the definition of “data” is in error. As pointed out above, “data” is information and a program (especially a device driver) is or contains information.

3. It is important to note that Appellant does not dispute that McGlone teaches loading device drivers from the CPU to the local controllers. This, along with a proper understanding of what a device driver contains, renders the arguments in section A above moot. McGlone clearly teaches loading device drivers from the CPU to the local microprocessor. Device drivers contain configuration data. If the device driver is loaded, the configuration data is also loaded.

C. Appellant argues that there would be no reason to make the proposed modification.

As pointed out above, there is really no “proposed modification”. The rejection merely

Art Unit: 3714

points out that the device drivers are obviously configuration data and posits a reason for using the preferred embodiment instead of the second embodiment. McGlone discloses a preferred embodiment in which configuration data (in the form of device drivers) are loaded to the reel controller by the CPU. This version does not have non-volatile memory. Yet even if there were a modification, such a modification would result in a reduction in the duplication of parts and in a cost savings. This is because RAM is significantly less expensive than ROM. As pointed out in the Advisory Action, Appellant has failed to provide any evidence concerning the relative costs for RAM and ROM chips. Therefore Appellant's allegation that there would be no cost savings is completely unsupported by evidence. It is merely a statement of Appellant's opinion.

Examiner has researched differences in RAM and ROM pricing and has discovered that RAM is significantly cheaper than ROM. While the Examiner does not know exactly what types of RAM and ROM might be used in construction of a reel controller, Examiner has discovered that 250 MB of Ram costs something in the neighborhood of \$50. In contrast, a single megabyte of ROM costs \$355. (See Appendix B.) This strongly suggests that there would be significant cost savings to be had by replacing the expensive ROM chips with cheaper RAM chips.

*In view of the foregoing, Examiner respectfully requests the rejection of claims 1-5, 7-23 and 27 be affirmed.*

III. Applicant argues that claims 15, 17, 19 and 22 should not have been rejected over McGlone. First, Appellant argues that McGlone fails to teach downloading configuration data to the local microprocessor of the reel controller from the CPU. As discussed immediately above,

Art Unit: 3714

this is not the case. McGlone's preferred embodiment loads configuration data to the local microprocessor of the reel controller from the CPU.

Appellant then argues that Examiner fails to establish a prima facie case of obviousness because McGlone fails to explicitly teach all of the limitations of the claim. Applicant is in error. If Appellant's argument were valid, it would never be possible to make a rejection under 35 USC §103(a) using a single reference. Yet such rejections are common, and, what is more, they are often appropriate. If a problem and solution are well known to the art, it is appropriate to apply that knowledge to a disclosure without use of another reference.

In this case, slot machines reel controllers require absolute accuracy. This is not only to meet the requirements of stringent regulation, but also to keep casinos from paying significant amounts of money. This is a well-known problem. McGlone teaches communications between the reel controller's local microprocessor and the CPU, but does not specifically teach the content of this communication. It is well known to double check critical data. One of the oldest rules of computer programming is that if a program writes a value (i.e., loads it into memory), the program should check to see that the value was correctly written. Checking data integrity is well known to the art and is imperative if a system is to work properly. Applicant's argument, if accepted, would result in the granting of a patent for following a standard computer programming practice.

*In view of the foregoing, Examiner respectfully requests the rejection of claims 15, 17, 19, and 22 be affirmed.*

IV. Appellant argues that claims 23 and 27 are improperly rejected. Applicant argues that McGlone teaches that checking the type of encoder is not necessary because McGlone does not



Art Unit: 3714

teach loading the configuration data to the reel controller. As discussed in connection with claims 1-5, 7-23 and 27, the preferred embodiment of McGlone's invention does teach loading the configuration data from the CPU to the reel controller. Thus it would certainly be necessary to determine which device driver to load in order to ensure the proper functioning of McGlone's device. Since all of Appellant's arguments are based on the wrong embodiment of McGlone's invention, they are moot.

*In view of the foregoing, Examiner respectfully requests the rejection of claims 23 and 27 be affirmed.*

V. Appellant argues that claim 28 is improperly rejected. Appellant argues that McGlone does not suggest determining the type of encoder present. As noted above, this is an error caused by using the wrong embodiment described in McGlone to formulate the argument. Appellant also argues that McGlone determines the device driver to load based on a series of numbers. This is true, but this is only after the master game controller determines what type of encoder present (Fig 7, 730, Col 16, 17-43). This is accomplished by querying the local controller. Note that in block 700, McGlone discloses running diagnostics to determine is the peripheral is running properly. This would include spinning the reels and testing the encoder. This would generate the best data concerning what type of encoder was actually present. Data in a table may be corrupt. It may be entered wrong. It may reflect hardware that was changed without updating the table. But data gathered from the device itself cannot have these problems. The easiest and best way to determine what type of encoder is actually physically present on the machine is to gather that information from the encoder itself by spinning the reels.

*In view of the foregoing, Examiner respectfully requests the rejection of claim 28 be affirmed.*

VI. Appellant argues that rejection of claim 29 was improper. Appellant argues that McGlone fails to teach transferring the table of step rates from the CPU to the reel controller. As pointed out above, this argument is based on the less preferred embodiment of McGlone's invention. As pointed out by the Appellant, McGlone does teach that the stepper motor rate instructions are loaded onto the reel controller. McGlone teaches that the reel controller issues low level commands to the stepper motor in response to high-level commands from the master gaming controller. The full paragraph that includes the portion quoted in Appellant's Brief (page 17) clearly describes this process:

The devices comprising the slot reel peripheral may be controlled directly by the master gaming controller 422 via a series of low-level instructions or indirectly by the master gaming controller via high-level instructions to the slot reel controller 402 which then sends out the low-level instructions. For example, to spin up the slot reel 420 from an initial non-rotating position and then to spin it down to a final position, the stepper motor 418 might require a series of low level instructions including charge the motor, initiate the first step, first delay period, initiate the second step, second delay period, initiate the third step, third delay period, initiate the fourth step, fourth delay period, perform the final step, and stop the motor. When the slot reel is accelerating, the length of time of each delay period between successive steps may decrease. When the slot reel is decelerating the length of time of each delay period between successive steps may increase. The step rate, which is a function of the length of time of each delay period between successive steps, may be based on a table stored in memory corresponding to the particular slot reel. When the master gaming controller directly controls the stepper motor, the master gaming controller would send the series of low-level instructions to the stepper motor. However, with a slot reel peripheral 400 containing a slot reel controller 402, the master gaming controller might send a high-level instruction to the slot reel controller 402 corresponding to a series of low-level instructions for a particular device. The slot reel controller 402 may interpret the high-level instruction and convert it to a series of low-level instructions. For the stepper motor example described above, the low-level commands, charge the motor, initiate first step, step at rate 1, step at rate 2, step at rate 3, step at rate 4, perform final step, and stop the motor, might be initiated by

Art Unit: 3714

the slot reel controller 402 after receiving a high-level instruction from the master gaming controller 422 like "move the slot reel 420 to position A." (Col 9, 23-57) (Emphasis Added)

Appellant's argument has clarified the meaning of claim 29 to the Examiner. Examiner now understands that McGlone also teaches an acceleration profile and deceleration profile. Therefore, it was not necessary to make reference to Sakamoto. However, Sakamoto is more explicit in its teaching than McGlone. Therefore, the rejection is still proper since both Sakamoto and McGlone teach an acceleration profile.

On page 6 of Appellant's specification an "acceleration profile" is defined as telling the reel "how to begin spinning". Sakamoto does this by having the CPU tell the stepper motor what rate and direction to spin at. McGlone's CPU issues a high level command that says, "move the slot reel 420 to position A". This tells the reel how to begin spinning – in a certain pre-programmed direction and at a certain pre-programmed rate. It also issues a deceleration profile telling the reel controller "how to stop spinning the reel". It tells the reel controller to stop spinning the reel so that it comes to rest at position A. Therefore, McGlone renders the claimed invention obvious.

*In view of the foregoing, Examiner respectfully requests the rejection of claim 29 be affirmed*

Art Unit: 3714

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,




Corbett B. Coburn III  
Examiner

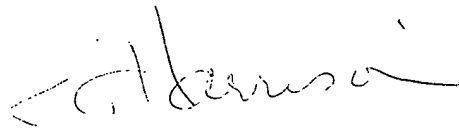
September 11, 2003

Conferees

Tom Hughes  
SPE



Jessica Harrison  
Primary Examiner



JESSICA HARRISON  
PRIMARY EXAMINER

JENKENS & GILCHRIST, P.C.  
225 WEST WASHINGTON  
SUITE 2600  
CHICAGO, IL 60606

# Appendix A

# MICROCOMPUTERS

Peter A. Sandon, IBM Microelectronics Division

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering Online*  
Copyright © 1999 by John Wiley & Sons, Inc. All rights reserved.  
Article Online Posting Date: December 27, 1999

[ARTICLE CONTENTS](#) [PREVIOUS](#) [NEXT](#)

## MICROCOMPUTER SOFTWARE

The information that controls the behavior of a computer is called software. It consists of both instructions and the data used by those instructions for decision-making. Software is often categorized as either an application program or system software. Applications are designed to be run by users to accomplish some task. System software, in particular the operating system (OS), is designed to supervise the execution of applications, and to provide services for those applications. Some programs, such as programming language translators - compilers, assemblers, interpreters - share characteristics of both application and system code.

### Application Programs

Microcomputers are most often used by a single user in an interactive mode. Many applications have been developed for microcomputers specifically aimed at this interactive style of computation. For example, what-you-see-is-what-you-get (WYSIWYG) word processors format text as it is input rather than through a postprocessing step. Spreadsheet programs calculate tabular data on-the-fly, providing immediate feedback for testing alternative hypotheses or investigating how a change in one parameter affects the values of other parameters. Image-processing programs allow interactive analysis and enhancement of image data. Media-editing applications support unlimited experimentation with cuts, joins, and special effects to obtain suitable sequences of audio and video streams. Games, educational applications, and desktop publishing programs are also designed around the interactive aspect of microcomputer use. Even applications development itself is well supported through integrated development environments in which program editors, compilers, and debuggers are combined to streamline program development. Of course, noninteractive applications, such as scientific (numeric) programs and data-processing programs - bookkeeping, inventory, database - are also available for microcomputers.

To run an application, it must have space allocated for it in memory for both the instructions and the data that it will use. The program and data are then loaded into memory and linked to (supplied with the actual memory location of) any dynamic libraries that it calls. The processor then branches to the first instruction in the application, and it begins to execute. While executing, if data or instructions are referenced that are not currently in memory, they must be moved into memory. If an application needs data from a hard disk, or prints a message to the screen, or checks to see if a key on the keyboard has been pressed, the corresponding I/O operation must be performed. All of these functions - managing memory space, loading applications, controlling I/O operations, among others - are performed by the processor executing instruction sequences that are part of the operating system (OS).

### Operating System

There are several major subsystems in an operating system, including the process scheduler, various resource managers (file system, I/O, memory), and the program loader. An application gains access to resources managed by the OS through calls to dynamic libraries. The OS, in turn, uses device drivers to provide control functions for specific I/O devices. In addition to supporting applications by providing

common functions that would otherwise have to be replicated in every application, these OS modules provide security to the system and its users. This is accomplished through the use of certain instructions and certain data areas in memory that can only be accessed by the operating system.

### ***Process Scheduler***

The process scheduler determines which of perhaps several available instruction streams, called runnable processes, should be executed next. Early microcomputer operating systems were single-tasking, meaning that there was only one runnable process at any given time. This process was either a command shell in the operating system waiting for user input, or an application that the user chose to run.

More recent operating systems allow nonpreemptive, or cooperative, multitasking. This means that multiple processes may be runnable at any given time, but once the scheduler chooses one to execute, that process executes until it completes. The operating system has no mechanism to stop it.

As with microcomputer hardware, operating systems for microcomputers have evolved and grown more complex, and have inherited functionality from main frames and minicomputers. The most recently developed microcomputer operating systems support preemptive multitasking. This means that multiple processes may be runnable, and that once a process starts to run, it may be suspended by the operating system at any time, to allow another process to run. This capability is particularly important for multiuser systems, where it provides time-sharing of the processor in such a way that each user has the impression that their application is progressing at a steady rate. However, it is also important in a single-user microcomputer, both to support particular styles of programming (multithreading), and to allow efficient and convenient background execution (e.g., spooling), at the same time that one or more interactive applications are running.

### ***Memory Manager***

Main memory is physically organized as a one-dimensional array of storage elements, each identified by its order in the array, called its address. All of the information used by the processor to do work, including both instructions and data, must be stored in main memory in order to be accessible to the processor. The memory manager must partition main memory so that each of the different software components that require this resource at any given time have the needed space. Among the different partitions required are those for base OS code and data, for applications code and data, and for dynamic libraries and device drivers.

Today's microprocessors provide hardware support for memory managers to implement a virtual memory. The idea is that the memory manager can behave as if it had a very large memory to work with, and each application has its own memory distinct from that of other applications. This simplifies the memory-management task. However, more space may be allocated in this very large virtual memory than is actually available in the physical memory. The virtual memory system, a combination of microprocessor hardware and OS code, solves this problem by moving information as needed between main memory and backing store. This gives the appearance of having a very large main memory.

### ***Dynamic Libraries***

Application programs request operating system services by calling library routines. Each of the services has associated with it an application programming interface (API), which defines the format the application must use to interact with the service. The API provides a level of abstraction between the application and the library. This allows the details of the library software or the hardware involved in the service to change, while the application software remains unchanged.

A library is simply a collection of software functions commonly used by applications. Dynamic libraries, also called shared libraries, are loaded into memory once and retained, so that any application that needs them can access them. Such a library function is dynamically linked to an application that references it when the application is loaded. This dynamic linking reduces the size of the application, and allows the library routine to change without a corresponding change to the application.

### ***Device Drivers***

Among the services that an operating system provides to an application program is I/O processing. When an application specifies that a particular data stream is to be written to the display, or that a new file should be created on the hard disk, or the next keystroke should be read in, operating system code is executed to perform the requested function.

The request from the application is abstract, in the sense that it is made independent of which particular device or even class of device will be involved in satisfying the request. The I/O manager has knowledge of different classes of devices, but does not have specific information on how to control every possible I/O device that might be attached to the microcomputer.

The device driver is the piece of code that does have device specific information. When a particular device is installed, the corresponding device driver software is installed as well. When the I/O manager gets a request to perform a particular function on a particular type of device, it passes the request to the appropriate device driver, which turns the request into the correct control sequence for that device.

### ***Booting the Computer***

RAM memory, used for most of main memory and caches, is volatile. That is, it loses its information whenever the power is turned off. When a microcomputer is first turned on, main memory has no information in it. In order for the operating system to load a program, the OS must already be in memory. But how does the operating system itself get loaded? In a reference to the expression "picking oneself up by the bootstraps," the process of getting the computer to bring itself to a state where it can run programs is called bootstrapping, or just booting.

The set of instructions for booting the computer, the boot code, is stored in ROM memory, a nonvolatile, nonwriteable form of IC memory. Since instructions in ROM cannot be changed, programs in ROM are often referred to as hard-wired, or hard-coded. Since boot code has this property of being hard-coded software, it is also referred to as firmware.

The boot code performs two functions. First, it checks the hardware to determine that enough of it is functional to begin loading the operating system. In particular, it exercises the basic functionality of the microprocessor, writes and reads the RAM memory to check for data errors, and tests the display adapter, disk drives, and keyboard to verify that they are operational. Second, the boot code loads the operating system. Although loading the operating system can be involved, the boot code itself need only get the process started. Once it locates the device from which the operating system is to be loaded (usually a hard disk, sometimes a CD-ROM, or even the LAN), the boot code loads a program from that device containing information needed to load other pieces of software. These, in turn, may take part in the loading of the rest of the operating system. In this way, the computer "picks itself up by its bootstraps."





[\[EEEE Home\]](#) [\[A to Z\]](#) [\[Subjects\]](#) [\[Search\]](#)

---

Copyright ©1999-2000 by John Wiley & Sons, Inc. All rights reserved.

# Appendix B

How can IT managers justify  
costs for deploying  
tablet PCs?

Click here for the answer

**Jupiterrese**  
Business Intelligence for Business

[Home](#) | [Windows](#) | [Win Drivers](#) | [Hardware](#) | [Downloads](#) | [Networking](#) | [News](#) | [Wireless](#)

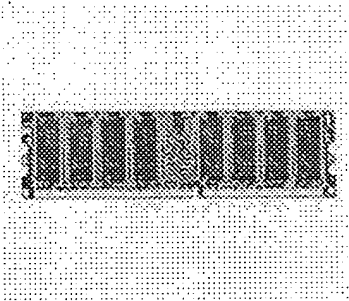
**HardwareCentral**  
Your source for in-depth computer hardware info.

Computers Desktops Digital Cameras Camcorders DVD Players Electronic  
Home Theater Laptops MP3 Players Office PDAs Software Telephones Wireless  
HOME Search for:  in Random Access Memory

Powered by **DeaTime**

Back to : [Computers](#) -> [Computer Memory](#) -> [Random Access Memory](#)

## Kingston 256 MB DDR SDRAM (KVR266X64C25/256) 266 MHz Bus Clock Rate Random Access Memory: Overview



[Read 1 review](#) | [Write a review at Epinions.com](#)

Kingston, the world's largest independent memory manufacturer, has over 2,000 memory products. In today's performance-driven environment, memory upgrades provide an easy, economical alternative to increase system performance. Every memory product Ki [Full Description >](#)

**Features:** 256 MB, DDR SDRAM, DIMM 184-pin, For: PC systems, Package Qty: 1 [Full Spec >](#)

Compare prices here:

Enter your ZIP CODE to calculate tax & shipping

Store Name	Store Rating	Base Price	+ Approx Tax* & Shipping	Total Price	Availability/Notes
	 <a href="#">read reviews (76)</a>	\$47.95	Enter ZIP CODE above to see the total price including tax & shipping		<ul style="list-style-type: none"> <li><b>IN STOCK!</b></li> <li>Usually ships in 2-3 days</li> </ul> <a href="#">Buy it at Gateway</a>
	 <a href="#">read reviews (91)</a>	\$44.40	Enter ZIP CODE above to see the total price including tax & shipping		<ul style="list-style-type: none"> <li><b>IN STOCK!</b></li> </ul> <a href="#">Buy it at Mwave</a> or call: 1-800-234...
	 <a href="#">read reviews (49)</a> <b>DEALTIME CERTIFIED</b>	\$48.00	Enter ZIP CODE above to see the total price including tax & shipping		<ul style="list-style-type: none"> <li><b>IN STOCK!</b></li> </ul> <a href="#">Buy it at accupc.com</a> or call: 1-866-286...
	 <a href="#">read reviews (54)</a>	\$50.40	Enter ZIP CODE above to see the total price including tax & shipping		<ul style="list-style-type: none"> <li><b>IN STOCK!</b></li> </ul> <a href="#">Buy it at PCNation</a> or call: 1-800-470...
	 <a href="#">read reviews (564)</a>	\$49.99	Enter ZIP CODE above to see the total price including tax & shipping		<ul style="list-style-type: none"> <li><b>IN STOCK!</b></li> </ul> <a href="#">Buy it at Amazon</a>

[See all 22 store offers \(range: \\$42.00 - \\$71.51\)](#)

BEST AVAILABLE COPY

Why are these stores listed?

\* State Tax Only. Does not include local taxes. Shipping costs are estimates. Please check merchant site for exact shipping costs.


## Kingston 256 MB DDR SDRAM (KVR266X64C25/256) 266 MHz Bus Clock Rate Random Access Memory Product Description

Kingston, the world's largest independent memory manufacturer, has over 2,000 memory products. In today's performance-driven environment, memory upgrades provide an easy, economical alternative to increase system performance. Every memory product Kingston offers is designed to help you get maximum performance at the best price to you.

## Kingston 256 MB DDR SDRAM (KVR266X64C25/256) 266 MHz Bus Clock Rate Random Access Memory Product Specs

### Key features

Capacity	256 MB
Technology	DDR SDRAM
Form Factor	DIMM 184-pin
Platform	PC

 [See full specs](#)

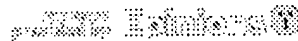
**BEST AVAILABLE COPY**

## Kingston 256 MB DDR SDRAM (KVR266X64C25/256) 266 MHz Bus Clock Rate Random Access Memory Product Reviews

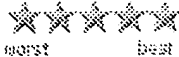


[Read 1 review](#) | [Write a review at Epinions.com](#)

### Selected ratings and reviews



#### AUTHOR'S RATING



#### If you look hard enough, you can find this for cheap

**By:** elffy  
July 19th, 2003

**Pros:** Price (read on, mcduff) Very High Quality RAM

**Cons:** Its not 512 meg! Might have to suffer dealing with a Rebate

**Review:** If you see it and are willing to deal with rebate headaches, its well worth it. Lifetime warranty. Big performance gain over old SDRAM. [More >>](#)

Copyright 2003 Jupitermedia Corporation All Rights Reserved.

[Legal Notices](#), [Licensing](#), [Reprints](#), & [Permissions](#), [Privacy Policy](#).  
<http://www.internet.com>

[Copyright © 2000-2003 DealTime Ltd.](#)

Prices are provided by the merchants. We assume no responsibility for accuracy of price information provided by merchants. However, we gladly field inquiries about pricing discrepancies and will bring them to the attention of the merchant in question. Write to us by [clicking here](#).

Product specifications are obtained from third parties, and while we make every effort to assure the accuracy of product information, we do not assume any liability for inaccuracies. Please bring any product information discrepancies to our attention by [clicking here](#).

Store ratings and product reviews are written and submitted by users of this site to assist you as you shop. They do not necessarily reflect our opinions. We take no responsibility for the content of ratings and reviews submitted by users.

How can IT managers justify costs for deploying tablet PCs?

Click here for the answer

**Jupiterresearch**  
Business Intelligence for Business

[Home](#) | [Windows](#) | [Win Drivers](#) | [Hardware](#) | [Downloads](#) | [Networking](#) | [News](#) | [Wireless](#)



Computers Desktops Digital Cameras Camcorders DVD Players Electronic  
Home Theater Laptops MP3 Players Office PDA's Software Telephony Video

HOME Search for:  in **Read Only Memory** HELP

Powered by DealTime

[Back to : Computers -> Computer Memory -> Read Only Memory](#)

## Intel 1 MB ROM (UPGLHE41MB) Read-Only Memory: Overview



Additional memory is always a great plus. You will experience better performance and higher speeds.

**Features:** 1 MB, ROM, For: PC, Unix systems, Package Qty: 1 [Full Spec >](#)

We found 1 match at 1 store



Enter your ZIP CODE to calculate tax & shipping

Store Name	Store Rating	Base Price	+ Approx Tax* & Shipping	Total Price	Availability/Notes
	 <a href="#">read reviews (72)</a> 	\$355.86	Enter ZIP CODE above to see the total price including tax & shipping	● <b>OUT OF STOCK</b>	<a href="#">Buy it at</a> <b>PC Connection</b> or call: <b>1-888-213...</b>

[Why are these stores listed?](#)

\* State Tax Only. Does not include local taxes. Shipping costs are estimates. Please check merchant site for exact shipping costs.

### Alternate Resources

#### [Computer Memory Upgrade](#)

Guaranteed Compatibility, Easy to Use Website, Free Shipping, No Tax  
[www.4AllMemory.com](http://www.4AllMemory.com)

## Intel 1 MB ROM (UPGLHE41MB) Read-Only Memory Product Description

Additional memory is always a great plus. You will experience better performance and higher speeds.

BEST AVAILABLE COPY

Copyright 2003 Jupitermedia Corporation All Rights Reserved.

[Legal Notices, Licensing, Reprints, & Permissions, Privacy Policy.](#)  
<http://www.internet.com>

Copyright © 2000-2003 DealTime Ltd.

Prices are provided by the merchants. We assume no responsibility for accuracy of price information provided by merchants. However, we gladly field inquiries about pricing discrepancies and will bring them to the attention of the merchant in question. Write to us by [clicking here](#).

Product specifications are obtained from third parties, and while we make every effort to assure the accuracy of product information, we do not assume any liability for inaccuracies. Please bring any product information discrepancies to our attention by [clicking here](#).

Store ratings and product reviews are written and submitted by users of this site to assist you as you shop. They do not necessarily reflect our opinions. We take no responsibility for the content of ratings and reviews submitted by users.

**BEST AVAILABLE COPY**